2025 年 8 月 20 日

SuperCon 2025 本選問題：洋上風力発電所の設置

1　目的

風力発電は再生可能エネルギーの中核を成しており，特に日本では図 1 に示すような洋上風力発電に大きな期待が寄せられている．日本は国土面積が狭く山地が多いため大規模な陸上風力発電の設置には限界がある一方，世界第 6 位の広さを誇る排他的経済水域を有し，その中には風況に恵まれた海域が数多く存在する．しかし，その多くが沿岸から急深となる地形のため，欧州などで一般的な水深 50 m 以下の着床式風車を設置しづらく，浮体式洋上風車の開発が急務である．浮体式であれば水深 200 m 級の海域にも対応でき，日本の約 608 GW と推定される莫大な洋上風力ポテンシャルを活かす鍵となり得る．

図 1　秋田港洋上風力発電所（秋田洋上風力発電株式会社）．出典：経済産業省 資源エネルギー庁「2025 年、日本の洋上風力発電〜今どうなってる？これからどうなる？〜」

洋上風力発電所の計画・設計では，風車（風力タービン）のレイアウト最適化と設置台数の最適化が不可欠である．洋上風力発電では，風車同士が近すぎるとそれらが相互に干渉し風向きの下流にある風車の発電量が低下してしまう（ウエイク現象による発電量の低下）ため，**互いに風を奪い合わない適切な間隔を保ちつつ，強風が得られる地点を狙って配置する**必要がある．また，洋上設置には巨額の初期投資を要するため，目標発電量を効率的に達成するための**最適な風車数**を見積もることが，コスト低減と経済性向上に大きく

寄与する．そこで本課題では，日本近海を想定し，デカルト座標系 (x, y) 上に，設置台数と配置位置を同時に最適化するモデルを構築する．日本近海の状況を反映し，すべての風車は浮体式（海底の水深制約なし）で海域内の任意の地点に設置できる前提とする．与えられた海域内で，**上限 $N_{\max}$ 台まで**の風力タービンを配置し，**総発電量を最大化**するとともに，**使用風車数を最小化**する最適解を探してみよう．

2　洋上風力発電モデル

本課題では，洋上風力発電所内の風車配置と発電量評価のために以下のモデル・仮定を用いる．

2.1　海域の表現

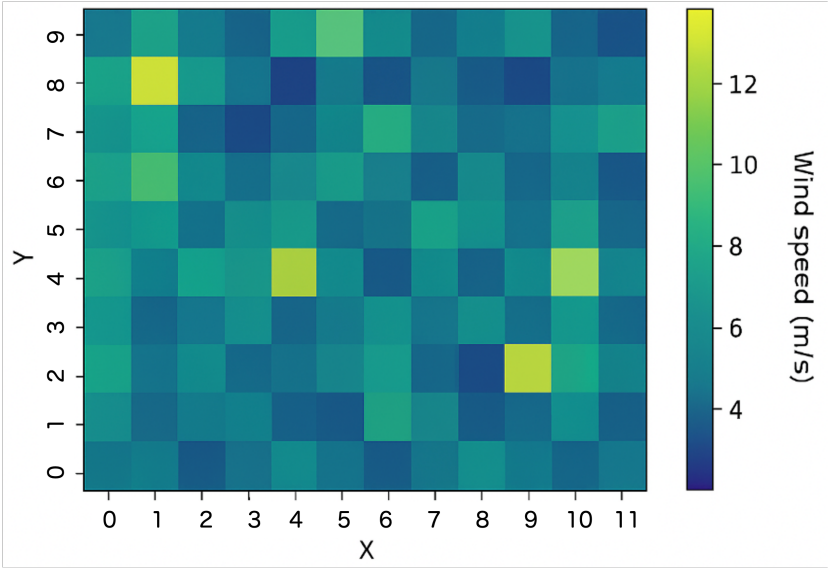

図 2　2 次元グリッドで表された海域の風速マップ．各セルにはそのポイントに風車を 1 基設置したときの発電ポテンシャルを表す基準風速 $W(x, y)$（単位：m/s）が与えられている．

風車の配置対象となる海域はサイズが $x \times y$ の 2 次元グリッドで表され，各グリッドセルは風車の配置可能なポイントを示す．セルの座標は (x, y) で表し，ここでは，$x = 0, 1, \ldots, X-1$，$y = 0, 1, \ldots, Y-1$ の整数グリッド座標を用いる（つまり，左下のセルの座標は $(0, 0)$，右上のセルの座標は $(X-1, Y-1)$ となる）．各セルにはそのポイントに風車を 1 基設置したときの発電ポテンシャルを表す基準風速 $W(x, y)$（単位：m/s）

が与えられており，それ以外の地形情報や海流データなどは含まれていない．図 2 は各セルに与えられた基準風速 $W(x, y)$ を可視化した風速マップを示している．

2.2 風車後流による風速低下

風車が風のエネルギーを取り出すと，その風車の下流には風速の遅い領域（後流）が発生する．この現象は「ウエイク現象」と呼ばれ，風下側の風車の発電量に直接的な影響を与える．風車近傍では大きな風速低下が見られる一方で，離れるほど周囲の風と混ざり徐々に風速は回復する．この後流の広がりと風速低下を定式化した簡単な解析モデルの一つが Jensen wake model（ジェンセンモデル）であり，本課題では風車間の影響計算にこの解析モデルを使用する．

ジェンセンモデルでは，風車後流の半径が下流に進むにつれて一定の拡大角で広がるものと仮定する．ある位置に風車を配置した場合，その風車から風下方向に距離 x 離れた点での後流半径 $r(x)$ は，風車ロータ半径 r_0 および後流の拡散度合いを表す物理量（後流の拡散係数）k を用いて次式で表される．

$$r(x) = r_0 + kx. \tag{1}$$

なお，k は地表の地形や大気の乱流強度によって通常変化するが，本課題では一律な定数（$k = 0.04$）として設定する．

風車直後の風速低下は風車のスラスト係数 C_t に依存する．C_t は風車が風から受ける「押す力」が，風の勢いの基準に対してどのくらいかを表す「単位のない割合（無次元の係数）」であり，典型的には 0.5 ～ 0.8 程度の値をとる（この値が大きいほど，風を強く減速させる）．ジェンセンモデルでは，風車直後の風速が風車を配置したポイントにおける自由風速 v_0（後流の影響を受ける前の風速 = 風車を配置したポイントにおける基準風速 W）に対してどの程度減少するかを C_t から次のように計算する．

$$v_{down,0} = v_0\sqrt{1 - C_t}. \tag{2}$$

つまり，風速の低下率（欠損率）D (Deficit) は

$$\begin{aligned} D &= 1 - \frac{v_{down,0}}{v_0} \\ &= 1 - \sqrt{1 - C_t} \end{aligned} \tag{3}$$

となる．この後流の速度欠損が距離とともに薄まっていくと考え，距離 x での風速 $v(x)$

は次式で表される．

$$v(x) = v_0 \left[1 - D \left(\frac{r_0}{r(x)} \right)^2 \right]. \tag{4}$$

この式は，後流断面の半径方向に均一な「トップハット型」の速度分布を仮定し，面積比 $(\frac{r_0}{r(x)})^2$ に応じて速度低下分が減衰するというモデルになっている．すなわち，距離が大きくなるほど $(\frac{r_0}{r(x)})^2$ 項が小さくなるため，$v(x)$ は風車を配置したポイントにおける自由風速 v_0 に近づく．そして，この式によって求められた $v(x)$ と距離 x のポイントにあらかじめ与えられている自由風速を用いることによって，距離 x での実効風速が計算される．

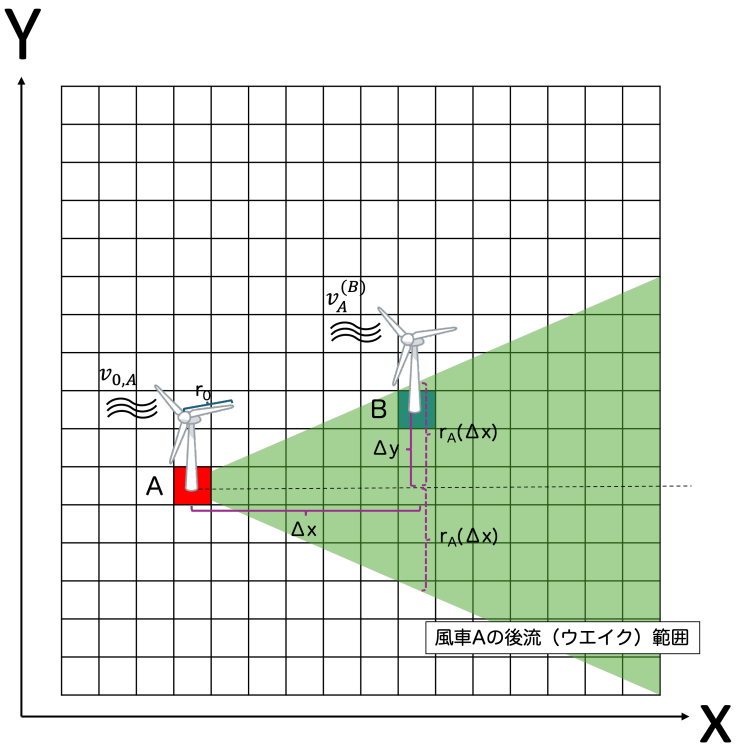

図 3　ジェンセンモデルにおける風車 A の後流（ウエイク）範囲.

本課題では風向きを一定と仮定し，**風は常に西から東（x 軸正方向）へ吹くものとする．**したがって, ある風車 A が風車 B よりも西側（$x_B > x_A$）にあれば, B は A の後流内に入る可能性がある．ウエイクの影響判定はジェンセンモデルの後流半径 $r_A(\Delta x) = r_0 + k\Delta x$（式 1）に基づいて行う．ここで，風車 A と風車 B の距離を

- $\Delta x = x_B - x_A$（東西方向の距離）

- $\Delta y = |y_B - y_A|$（南北方向の距離）

としたとき，$\Delta y \leqq r_A(\Delta x)$ を満たすのであれば，図 3 に示すように B は A のウエイク範囲内にあるものとみなす．その場合，A から B に届く風速は式 4 により，

$$v_A^{(B)} = v_{0,A}\left[1 - D\left(\frac{r_0}{r_A(\Delta x)}\right)^2\right] \tag{5}$$

で与えられる．

さらに B が複数の上流風車（A，C，D …）のウエイクに重なる場合，それぞれの風車からの「速度欠損量」を二乗和の平方根（Root Sum of Squares, RSS）で合成する．これは，複数のウエイクが重なっているポイントの風速が複数のウエイクの影響でどの程度遅くなっているのかを物理的に無理のない範囲で見積もるための近似である．単純な足し算（線形和）にすると，例えば 10 % の遅れ + 10 % の遅れ = 20% と合成するが，現実には空気は混ざって再び速くなろうとするので，ここまで大きくは落ちにくい（つまり，線形和だと速度欠損量が過大評価されやすい）．一方，RSS だと，10 % と 10 % は $\sqrt{0.1^2 + 0.1^2} \approx 0.141$ (14 %) になり，ウエイクが重なれば速度欠損量は当然増えるが，線形和と比べて「やり過ぎない」合成になるため，実測に近い減速を数学的に見積りやすい．RSS に基づくと，B での実効風速 $v_{\mathrm{eff},B}$ は，

$$v_{\mathrm{eff},B} = v_{0,B} - \sqrt{\sum_{j\in\text{ 上流風車}} \left(v_{0,B} - v_j^{(B)}\right)^2} \tag{6}$$

で与えられる．上流風車がなければ欠損和がゼロになり，$v_{\mathrm{eff},B} = v_{0,B}$（元の自由風速へ回復），複数のウエイクが重なれば欠損量が RSS 合成で累積されていく．ただし，欠損量がどれ程多かったとしても，実効風速 $v_{\mathrm{eff},B}$ の下限は 0[*1] とする．各風車の受ける実効風速はこのようにして求められ，それらの値は各風車の発電出力の算出に用いられる．

2.3 風車の発電出力特性

本課題では，風車の発電出力はその風車に到達する風速（実効風速 v_{eff}）のみによって決まるものとし，設置した風車の向きをはじめとしたその他の要因は考慮しない．風車の出力はパワーカーブ（出力曲線）と呼ばれる風速と出力の関係グラフで示される．典型的

[*1] 風速は「空気が単位時間に進む速さ」と定義されるスカラー量（速度ベクトルの大きさ）であるため，定義上 0 以上の非負の値をとる．

な風力発電用風車では，風速がある程度高まると出力が急激に上昇し，定格出力に達した後は制御によって出力が頭打ち（一定）になり，さらに非常に強い風では安全のため停止する．具体的には，表 1 に示すような 4 段階に分かれる．

表 1　パワーカーブの区間別出力特性

区間	風速範囲	出力の挙動
カットイン	$v < v_{\mathrm{ci}}$	風速不足で回転せず，出力 $P = 0$
起動 〜 定格	$v_{\mathrm{ci}} \leqq v < v_{\mathrm{r}}$	風速に応じて出力が急増
定格 〜 カットアウト	$v_{\mathrm{r}} \leqq v < v_{\mathrm{co}}$	定格制御により出力を一定 $P = P_{\mathrm{r}}$ に維持
カットアウト	$v \geqq v_{\mathrm{co}}$	安全停止のため回転を停止し，出力 $P = 0$

- カットイン風速 v_{ci}：風車が最低限回転を始める風速.
- 定格風速 v_{r}：風車が最大出力（定格出力）に達する風速.
- 定格出力 P_{r}：風車の最大出力.
- カットアウト風速 v_{co}：強風時に安全停止する風速.

本課題では高校数学の範囲内で扱いやすいよう，次のような折れ線近似でパワーカーブを定義する．

$$P(v) = \begin{cases} 0, & v < v_{\mathrm{ci}}, \\ P_{\mathrm{r}} \dfrac{v - v_{\mathrm{ci}}}{v_{\mathrm{r}} - v_{\mathrm{ci}}}, & v_{\mathrm{ci}} \leqq v < v_{\mathrm{r}}, \\ P_{\mathrm{r}}, & v_{\mathrm{r}} \leqq v < v_{\mathrm{co}}, \\ 0, & v \geqq v_{\mathrm{co}}. \end{cases} \tag{7}$$

例えば，v_{ci} を 3.0 m/s，v_{r} を 12.0 m/s，P_{r} を 2000 kW（2 MW），v_{co} を 25.0 m/s としたときのパワーカーブは図 4 のように図示できる．

すなわち，洋上風力発電所の総発電量は，各風車 i が受ける実効風速 $v_{\mathrm{eff},i}$ に対して個別に $P(v_{\mathrm{eff},i})$ を折れ線近似で定義したパワーカーブに基づいて計算し，全風車の合計をとることによって算出される．

$$P_{\mathrm{total}} = \sum_{i=1}^{\text{配置台数}} P(v_{\mathrm{eff},i}) \tag{8}$$

この値を本課題の評価軸の第一指標とする．

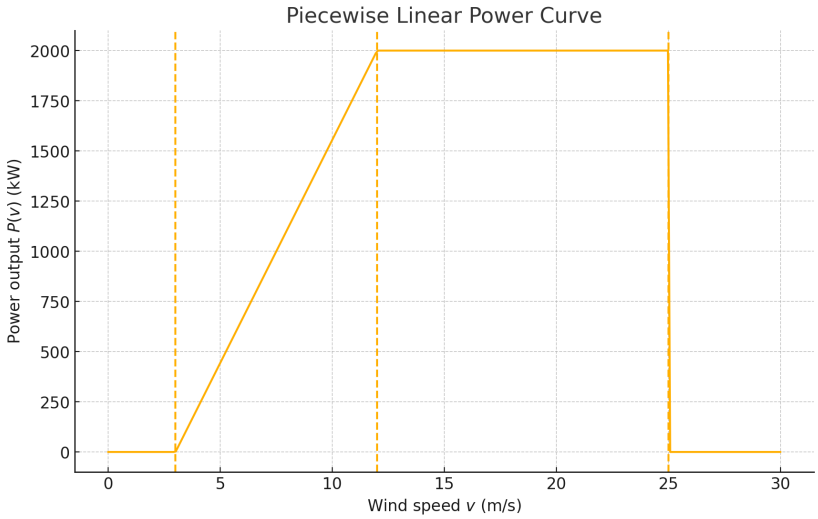

図 4　折れ線近似で定義したパワーカーブ．図中の破線は左からカットイン（$v_{\mathrm{ci}} = 3$ m/s），定格（$v_{\mathrm{r}} = 12$ m/s），カットアウト（$v_{\mathrm{co}} = 25$ m/s）を示している．

2.4　モデルの実行フロー

洋上風力発電モデルでは，次のように計算が実行される．

1. **入力読み込み**

   テキストファイルから以下を読み込む：

   - 海域サイズ X, Y（横 $\times$ 縦 のセル数．本課題では，**1 セルの長さを 20 m と定義する**）
   - 最小間隔 $D_{\min}$（安全や干渉防止のために空けなければいけない風車同士の最低距離）
   - 最大設置可能台数 $N_{\max}$
   - 風速マップ $W(x, y)$（各セルにおける基準風速）

   風向きは西から東（$+x$ 方向）と固定する．

2. **配置候補の絞り込み**

   風車同士が最小間隔 $D_{\min}$ を必ず満たすように，各配置候補セル (x_i, y_i) の組み合わせを生成・検査し，

   $$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \geqq D_{\min} \quad (\forall i \neq j)$$

   を満たすもののみを有効な配置パターンとして扱う．

3. **自由風速のセット**
   各配置候補セル (x_i, y_i) について，

$$v_{0,i} \;=\; W(x_i, y_i)$$

   を「自由風速」として取得.
4. **上流風車リストアップと後流判定**
   風車 i について，同チーム内で $x_j < x_i$ の全風車 j を上流風車として抽出し，

$$\Delta x = x_i - x_j, \quad \Delta y = |y_i - y_j|$$

   を計算する．式 1 で求めた後流半径 $r_j(\Delta x) = r_0 + k\,\Delta x$ と比較し，

$$\Delta y \leqq r_j(\Delta x)$$

   のとき影響範囲内と判定する.
5. **後流風速の計算と実効風速への反映**
   式 4 に基づいて，影響範囲内の各上流風車 j からの後流風速

$$v_j^{(i)} = v_{0,j}\left[1 - \left(1 - \sqrt{1 - C_t}\right)\frac{r_0^2}{(r_0 + k\,\Delta x)^2}\right]$$

   を計算し，欠損量 $\delta_j = v_{0,i} - v_j^{(i)}$ を求める．これらを RSS（Root Sum of Squares）で合成して，

$$v_{\mathrm{eff},i} = v_{0,i} - \sqrt{\sum_j \left(v_{0,i} - v_j^{(i)}\right)^2}$$

   として「実効風速」を得る（式 6）.
6. **出力計算**
   実効風速 $v_{\mathrm{eff},i}$ を式 7 のパワーカーブ $P(v)$ に代入し，

$$P_i = P\left(v_{\mathrm{eff},i}\right)$$

   を各風車の出力とする.
7. **総合評価**

- 総発電量：　$P_{\mathrm{total}} = \sum_{i=1}^{\text{配置台数}} P_i$（式 8）
- 使用台数：　実際に配置した風車の台数

- 実行時間：　プログラム実行に要した時間

本課題では，以下の優先順位で評価を行う：
(a) 総発電量を最大化
(b) 使用台数を最小化
(c) 実行時間を短縮

3　問題

Section 2 の洋上風力発電モデルを実行するプログラムを作成して，上限 $N_{\max}$ 台までの風力タービンを問題として与えられる海域内に適切に配置し，総発電量 P_{total} を最大化するとともに使用風車数を最小化せよ．

4　ルール

- 作成したプログラムで問題を 10 問解く．実行時間の上限は全体で 10 分とする．
- プログラムの入力として与えられるのは，10 個の問題に関するデータである．それぞれの問題については以下の値から構成されており，これらの値は問題ごとに異なる．
  - 海域サイズ X, Y：海域は X(横) $\times$ Y(縦) のセルからなる 2 次元グリッドで表される．本課題では，**1 セルの長さを 20 m と定義する．**
  - 最小間隔 $D_{\min}$：安全や干渉防止のために空けなければならない風車同士の最低距離（単位：セル数）．設置候補点はセル中心の整数格子点 $\mathcal{G} = \{0, 1, \ldots, X-1\} \times \{0, 1, \ldots, Y-1\}$（0-origin）とし，任意の 2 基 $(i_1, j_1), (i_2, j_2) \in \mathcal{G}$ のユークリッド距離 $d = \sqrt{(i_1 - i_2)^2 + (j_1 - j_2)^2}$ が $d < D_{\min}$ **にならない**（すなわち $d \geqq D_{\min}$ を満たす）ように配置する．例えば $D_{\min} = 2$ のとき，任意の 2 風車は 2 セル以上離れていなければならない．[*2]
  - 風車設置可能台数 $N_{\max}$：その問題で設置可能な風車の台数．本課題における $N_{\max}$ の最大値は，$[0, X-1] \times [0, Y-1]$ において $D_{\min}$ を保ちながら直交格

[*2] ユークリッド距離は二点間の直線距離で，ピタゴラスの定理により定義される．

子上に配置できる最大設置台数の上限であり，

$$s = \lceil D_{\min} \rceil, \qquad N_{\max}^{\mathrm{ub}} = \left\lceil \frac{X}{s} \right\rceil \cdot \left\lceil \frac{Y}{s} \right\rceil$$

とする．よって実際に用いる $N_{\max}$ は $0 < N_{\max} \leqq N_{\max}^{\mathrm{ub}}$ を満たす値に設定する．[*3]

– 風速マップ $W(x, y)$：X（横）$\times$ Y（縦）の各セルにおける基準風速（単位：m/s）が定義されている．入力は **行優先（*row-major*）** で，$y = 0, \ldots, Y-1$ の順に Y 行，各行は $x = 0, \ldots, X-1$ の順に X 個の実数を**空白（スペース／タブ／改行）** 区切りで与える．複数の空白や改行が連続してもよい（数値の前の空白は読み飛ばされる）．

- **10 個の問題に関するデータは，提供されたヘッダーファイル `sc25.h` に含まれる `SC25_input(const char *file_path)` 関数でプログラムに読み込ませること．この関数はプログラム中において一度のみ使用でき，何度も使用してはならない．** この関数を一度だけ呼び出すことで，10 問分の問題データが全てプログラムに読み込まれる．
- 作成するプログラムは総発電 P_{total} および風車の使用台数を算出する．**P_{total} は IEEE 754-2019 準拠の倍精度（binary64）で計算すること**．それ以外のレイアウト探索等には任意の表現（単精度・半精度・整数・固定小数点・FMA 活用など）を使って構わない．
- **総発電 P_{total}・風車の使用台数・レイアウト情報が本選問題における解答になり，これをヘッダーファイル `sc25.h` に含まれる `SC25_output(int i_prob, int sel_count, int sel_x[], int sel_y[], double P_total)` 関数を呼びだして出力すること．なお，本関数による解答の出力は各問題ごとに逐次（単一スレッドかつ単一プロセス）で行うこと．**
- **ヘッダーファイル `sc25.h` の書き換えは一切認めない．**
- **ソースコードは必ずひとつのファイルにまとめること．複数のファイルへの分割は一切認めない．** すなわち，ヘッダーファイル `sc25.h` を `#include` する単一ソースでなければならないので，ソースコードは `C / C++ / CUDA C++`（`.c / .cc / .cpp / .cu`）のいずれかで実装する必要がある．
- プログラムはタイムアウトで強制終了させるので終了処理は書かなくて良い．それまでに結果を問題ごとに何度出力しても構わないが，制限時間内に出力されたそれ

[*3] 天井関数 $\lceil x \rceil$ は「x 以上の最小の整数」を返す関数である．

ぞれの問題の最後の結果が審査に使われる．

- 問題には問題番号 0, 1, 2,..., 9 が割り振られているが，出力の順は問題番号の順にしたがう必要はない．また，10 問全ての解答が提出されていなくても，解答が提出された問題について審査する．
- **コンテストの審査対象となるプログラムはひとつのみ．単一ソースのファイルを複数提出することは一切認めない．また，`bash` コマンドで実行できるコンパイル用・実行用スクリプトをソースコードと併せて提出すること．`Makefile` の使用は不可．**

5 勝利条件

- 各問について，
  - 使用した風車の台数が $N_{\max}$ に収まり，
  - どの風車も $D_{\min}$ 以上の間隔を保ったまま配置され，
  - P_{total} の計算が正しく行われているか，

  を審査委員会が用意した検証用プログラムで検算し，**台数・間隔・P_{total} 照合のいずれか不一致となった解答は「未提出」とみなす．未提出の問題のポイントは 0 点，順位付けの対象外とする．**
- 各問について，P_{total} が大きいものから順に順位をつけ，
  - 1 位：5 点
  - 2 位：4 点
  - 3 位：3 点
  - 4 位：2 点
  - 5 位：1 点
  - 6 位以下：0 点

  のように点数化する．P_{total} が同一の場合は同順位・同ポイント（次の順位はスキップ）とする．10 問の点数を合計して総合順位をつけ，点数が等しい時には以下の基準で順位付けを行う．
  1. 有効解の件数の多い方（未提出の解答が少ない方）を上位とする．
  2. 全問題で使用したタービン本数の合計が少ないチームを上位とする．
  3. 全問題で使用したタービン本数の合計が等しい場合は，総計算時間（下記の `SC25_output` 関数で出力される「計算開始からの経過時間」のうち，制限時間

内に出力された最後のもの）が短いチームを上位とする．

6　ヘッダーファイル

配布するコンテスト専用ヘッダーファイル sc25.h に以下のように必要な定数・変数・配列・関数が定義されている．sc25.h は，東京科学大学の Box 上の「本選問題」フォルダからダウンロードできる．

```c
#ifndef INCLUDE_GUARD_SC25_H
#define INCLUDE_GUARD_SC25_H

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// The number of problems
#define SC25_NPROB 10

// Euclidean distance
#define SC25_DIST(x1, y1, x2, y2) \
  (sqrt(((x1) - (x2)) * ((x1) - (x2)) + ((y1) - (y2)) * ((y1) - (y2))))

// Jensen model's parameters
static const double r0 = (40.0 / 20.0);  // = 2.0 cells (1 cell = 20 m)
static const double k = 0.04;
static const double Ct = 0.8;

// The power curve's parameters
static const double v_ci = 3.0;
static const double v_r = 12.0;
static const double v_co = 25.0;
static const double P_r = 2000.0;

// Buffer for reading data for each problem
static int SC25_X[SC25_NPROB];
static int SC25_Y[SC25_NPROB];
static double SC25_Dmin[SC25_NPROB];
static int SC25_Nmax[SC25_NPROB];
static double *SC25_W_flat[SC25_NPROB];

// A variable for storing the timestamp when all problems have been read
static struct timespec SC25_ts_start;

static double SC25_time_spent(void) {
  struct timespec ts;
  clock_gettime(CLOCK_MONOTONIC, &ts);
  double time_spent =
      ((double)(ts.tv_sec) + (double)ts.tv_nsec * 1e-9) -
      ((double)(SC25_ts_start.tv_sec) + (double)SC25_ts_start.tv_nsec * 1e-9);
  return time_spent;
}

static void SC25_input(const char *file_path) {
  int my_id = 0;
#ifdef MPI_VERSION
  MPI_Comm_rank(MPI_COMM_WORLD, &my_id);
#endif
```

```
  if (!file_path || !*file_path) {  // argv[argc - 1] is NULL or empty string.
    if (my_id == 0)
      fprintf(stderr,
              "./<your_solver> <problems.txt> / mpirun -np N ./<your_solver> "
              "<problems.txt>\n");
#ifdef MPI_VERSION
    MPI_Abort(MPI_COMM_WORLD, 1);
#endif
    exit(EXIT_FAILURE);
  }
  FILE *fp = NULL;
  if (my_id == 0) {
    fp = fopen(file_path, "r");
    if (!fp) {
      perror("SC25_input: fopen");
#ifdef MPI_VERSION
      MPI_Abort(MPI_COMM_WORLD, 1);
#endif
      exit(EXIT_FAILURE);
    }
  }
  for (int p = 0; p < SC25_NPROB; p++) {
    if (my_id == 0) {
      if (fscanf(fp, "%d %d %lf %d", &SC25_X[p], &SC25_Y[p], &SC25_Dmin[p],
                 &SC25_Nmax[p]) != 4) {
        fprintf(stderr, "SC25_input: Input format error (Problem %d)\n", p);
#ifdef MPI_VERSION
        MPI_Abort(MPI_COMM_WORLD, 1);
#endif
        exit(EXIT_FAILURE);
      }
    }
#ifdef MPI_VERSION
    MPI_Bcast(&SC25_X[p], 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Bcast(&SC25_Y[p], 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Bcast(&SC25_Dmin[p], 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
    MPI_Bcast(&SC25_Nmax[p], 1, MPI_INT, 0, MPI_COMM_WORLD);
#endif

    // Allocate memory space for wind speed map
    SC25_W_flat[p] = (double *)malloc(sizeof(double) * (size_t)SC25_X[p] *
                                      (size_t)SC25_Y[p]);
    if (!SC25_W_flat[p]) {
      fprintf(stderr, "SC25_input: Memory allocation failed (Problem %d)\n", p);
#ifdef MPI_VERSION
      MPI_Abort(MPI_COMM_WORLD, 1);
#endif
      exit(EXIT_FAILURE);
    }

    // Read wind speed map
    if (my_id == 0) {
      for (int y = 0; y < SC25_Y[p]; y++) {
        for (int x = 0; x < SC25_X[p]; x++) {
          size_t idx = (size_t)y * (size_t)SC25_X[p] + (size_t)x;
          if (fscanf(fp, "%lf", &SC25_W_flat[p][idx]) != 1) {
            fprintf(
                stderr,
                "SC25_input: Wind speed map read error (Problem %d, %d, %d)\n",
                p, x, y);
#ifdef MPI_VERSION
            MPI_Abort(MPI_COMM_WORLD, 1);
#endif
            exit(EXIT_FAILURE);
          }
```

```
        }
      }
    }
#ifdef MPI_VERSION
    MPI_Bcast(SC25_W_flat[p], SC25_X[p] * SC25_Y[p], MPI_DOUBLE, 0,
              MPI_COMM_WORLD);
#endif
  }
  if (my_id == 0) {
    fclose(fp);
    clock_gettime(CLOCK_MONOTONIC, &SC25_ts_start);
  }
}

static void SC25_output(int i_prob, int sel_count, int sel_x[], int sel_y[],
                        double P_total) {
  int my_id = 0;
#ifdef MPI_VERSION
  MPI_Comm_rank(MPI_COMM_WORLD, &my_id);
#endif
  if (my_id == 0) {
    double elapsed = SC25_time_spent();

    char fname[64];
    snprintf(fname, sizeof(fname), "layout_t_p%02d.txt", i_prob);

    FILE *fp = fopen(fname, "w");
    if (!fp) {
      fprintf(stderr, "SC25_output: cannot open %s for write\n", fname);
    } else {
      fprintf(fp, "X=%d Y=%d Dmin=%lf Nmax=%d (Problem %d)\n", SC25_X[i_prob],
              SC25_Y[i_prob], SC25_Dmin[i_prob], SC25_Nmax[i_prob], i_prob);
      for (int i = 0; i < sel_count; i++) {
        fprintf(fp, "%d %d\n", sel_x[i], sel_y[i]);
      }
      fclose(fp);
    }
    printf(
        "elapsed_s=%8.3f  problem=%02d  used=%4d  power_kW=%10.1f  file=%s\n",
        elapsed, i_prob, sel_count, P_total, fname);
    fflush(stdout);
  }
}

static void SC25_finalize(void) {
  for (int p = 0; p < SC25_NPROB; p++) {
    free(SC25_W_flat[p]);
  }
}

#endif  // INCLUDE_GUARD_SC25_H
```

以上のようなヘッダーファイル sc25.h を

```
#include <math.h>
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
```

```
#include "sc25.h"
```

のようにインクルードして使う．sc25.h は最後にインクルードすること．また，MPI を使わない時は mpi.h をインクルードしないようにすること．繰り返すが，**ヘッダーファイルの書き換えは一切認めない．**

1. **SC25_input(const char *file_path) 関数はプログラム中で最初の実行文とすること (MPI を使用するときには，最初の実行文を MPI_Init(&argc, &argv) にする必要があり，SC25_input() はその直後の実行文とする)．これ以前に実行文を記述してはいけない．**SC25_input() を実行することにより必要なデータがコマンドライン引数で指定したテキストファイルから読み込まれる．**SC25_input() 関数は一度だけ呼び出すこと．**
2. **SC25_output(int i_prob, int sel_count, int sel_x[], int sel_y[], double P_total) 関数を呼ぶことにより i_prob 番目 ($\in \{0, 1, 2, .., N_{\text{prob}} - 1\}$) の問題に対する解答が出力される．SC25_output() 関数以外の方法で出力してはならない．**SC25_output() 関数は何度呼んでも良い．
3. SC25_time_spent() 関数は時間計測用に使っても構わない．

7 サンプルプログラム

サンプルプログラムとそのコンパイル用・実行用スクリプトは東京科学大学の Box 上の「本選問題」フォルダからダウンロードできる．以下のファイルが「サンプルプログラム」フォルダに格納されている．

- offshore_wind_seq.c：シングルスレッドで動作する CPU 実装．レイアウトを決定するためのアルゴリズムは貪欲法．配置済みタービン集合に対して，各セルの発電量スコアを評価し，スコアが最大となるセルにタービンを新たに配置していく．
- offshore_wind_mpi.c：複数の問題ケースを MPI で並列に解くプログラム．rank 0 が入力と最終出力を担当し，各 rank は割当てケースを offshore_wind_seq.c と同じ貪欲法で解いて結果を rank 0 に送る．
- offshore_wind_acc.c： OpenACC で offshore_wind_seq.c を GPU 化した版．貪欲探索の重い部分（全セル評価）を並列化して加速する．並列化に伴う浮動小数点の加算順序や「最大スコア同点」のタイブレークの違いにより，逐次版と配

置が完全一致しないことがあるが，意味論（「利得 > 0 を貪欲に選ぶ」）は一致している．多くのケースでは総発電量や使用タービン数は近い値に収束しやすいが，必ず差が小さいとまでは保証しない．

- `compile.sh`： コンパイル用のスクリプト例．コマンドラインから `bash compile.sh` を実行すると，サンプルプログラムが `offshore_wind_seq.c` → `offshore_wind_mpi.c` → `offshore_wind_acc.c` の順にコンパイルされる．
- `run.sh`：プログラム実行用のスクリプト例．コマンドラインから `bash run.sh <problem-input-file>`を実行すると，`compile.sh` によってコンパイルされたプログラムが `<problem-input-file>` で指定した問題データを読み込んで，総発電量・風車の使用台数・風車配置のレイアウトを出力する．

8 問題サンプル

問題サンプルは東京科学大学の Box 上の「本選問題」フォルダからダウンロードできる．「問題サンプル」のフォルダには全部で 20 個のテキストファイルがあり，各ファイルには 10 問の問題データが格納されている．なお，全 20 ファイルは，easy×10 / standard×7 / hard×3 というようにカテゴライズされている．3 つのカテゴリの概要は以下の通りである．

- `probs_easy_{01..10}.txt`：
  - 海域のサイズは小さめ ～ 中くらいのサイズで，なめらかな風の分布となっている．
  - まずは手を動かして，スーパーコンピュータのプログラミングに慣れるためのデータセットという位置づけ．
- `probs_standard_{01..07}.txt`：
  - 海域のサイズは中 ～ やや大きいめで，風の分布は穏やかな場 + 模様（縞・ホットスポット等）の混合．
  - 計算量が増える等の，探索の難しさが一段階上がるので，アルゴリズムや GPU 実装の最適化の効果を見るためのベンチマークとするのが良い．
- `probs_hard_{01..03}.txt`：
  - 海域のサイズは大きめで，最小間隔などの制約がやや厳しめ．
  - ウエイクの重なりが起きやすく，探索の難易度が高いベンチマークなので，実装の堅牢さと最適化の工夫が問われる．

これらのテキストファイルをコマンドライン引数を介してプログラムに渡せば，そのファイルに含まれる問題データ 10 問分が全て読み込まれる．例えば，Section 7 でサンプルをコンパイルした後に，`bash run.sh probs_easy_01.txt` とすれば `probs_easy_01.txt` に格納されている 10 問分の問題データを全て読み込み，風車配置の探索を実行する．

9 スーパーコンピュータ

本選では，東京科学大学のスーパーコンピュータ TSUBAME4.0 を用いる．出場者は，NVIDIA H100 GPU (SXM5 版) が 4 基搭載された計算ノードを 1 台使用できる．具体的な使い方は TSUBAME4.0 利用法説明スライドや Web で公開されている「TSUBAME4.0 利用の手引き」を参考にすること．

- 現在はさまざまなやり方（例：CUDA，OpenACC，Standard Language Parallelism）で GPU コンピューティングを実現できる．NVIDIA 社の資料を見ながら，色々試行錯誤してみてほしい．
- GPU を 1 つだけしか使わないのはもったいない．計算ノードに搭載された全ての GPU をフル活用して最適解を求めてみよう．マルチ GPU のプログラミングについても NVIDIA 社の資料を参考にしてほしい．

10 コンパイル方法

コンパイルはログインノード上ではなく，計算ノード上で行うこと (ログインノードには GPU ドライバがないので，GPU コードのコンパイルは失敗する)．Environment Modules（`module` コマンド）を使用することでコンパイラや `MPI` 環境の切り替えを行うことができる．`module` コマンドについての説明は，TSUBAME4.0 利用法説明スライドや Web で公開されている「TSUBAME4.0 利用の手引き」に記載されているので必ず目を通しておくこと．また，GPU コードのコンパイルは NVIDIA の資料を参照すること．その後，Section 7 で述べたサンプルプログラムを確認すれば，`module` コマンドをどのように使用し，プログラムをコンパイルしているのかを掴めるはずである．

11 提出物

各チームは以下の 3 点を提出すること．`XX` はチームの番号である．

- プログラムのソースコード 1 点：`sc25teamXX.[c|cc|cpp|cu]`
- コンパイル用スクリプト 1 点：`compile-sc25teamXX.sh`
- 実行用スクリプト 1 点：`run-sc25teamXX.sh`

ソースコードは必ずひとつのファイルにまとめること．複数のファイルへの分割は不可．すなわち，ヘッダーファイル `sc25.h` を `#include` する単一ソースでなければならないので，提出できるソースコードは `C / C++ / CUDA C++ (.c / .cc / .cpp / .cu)` のいずれかで実装されたものなら可とする．また，単一ソースのファイルを複数提出することは一切認めない．提出できるプログラムはひとつのみである．

コンパイル用および実行スクリプト用は `bash` コマンドで実行できるものであること．`Makefile` の使用は認めない．スクリプトの書き方は Section 7 で述べたサンプルを参考にすること．

上記 3 点のファイルを締切までに

```
/gs/bs/tge-sc2025tXX/submit
```

へ提出すること．なお，**提出前は慌てず，提出すべきものを提出すべき締切までにキチンと提出して頂きたい．**もし，

- 上記の要件を満たさない提出物（例：単一ソースのファイルを複数提出した）だった
- 提出されたコンパイル用スクリプトでソースコードのコンパイルができなかった
- 提出された実行用スクリプトでプログラムが動作しなかった

場合は，**審査不能により当該チームは無条件に失格**となるので十分に気をつけること．